



Calhoun: The NPS Institutional Archive

Theses and Dissertations

Thesis Collection

1963

Programming a remote monitor and display system for a time sharing computer.

Griffith, Webster.

Monterey, California: U.S. Naval Postgraduate School

<http://hdl.handle.net/10945/12900>



Calhoun is a project of the Dudley Knox Library at NPS, furthering the precepts and goals of open government and government transparency. All information contained herein has been approved for release by the NPS Public Affairs Officer.

**Dudley Knox Library / Naval Postgraduate School
411 Dyer Road / 1 University Circle
Monterey, California USA 93943**

<http://www.nps.edu/library>

NPS ARCHIVE
1963
GRIFFITH, W.

PROGRAMMING A REMOTE MONITOR
AND DISPLAY SYSTEM FOR A
TIME SHARING COMPUTER
WEBSTER GRIFFITH

LIBRARY
U.S. NAVAL POSTGRADUATE SCHOOL
MONTEREY CALIFORNIA

DUDLEY KNOX LIBRARY
NAVAL POSTGRADUATE SCHOOL
MONTEREY CA 93943-5101

PROGRAMMING
A REMOTE MONITOR AND DISPLAY SYSTEM
FOR A
TIME SHARING COMPUTER

* * * * *

Webster Griffith

PROGRAMMING
A REMOTE MONITOR AND DISPLAY SYSTEM
FOR A
TIME SHARING COMPUTER

by
Webster Griffith
//

Submitted in partial fulfillment of
the requirements for the degree of

MASTER OF SCIENCE
IN
ENGINEERING ELECTRONICS

United States Naval Postgraduate School
Monterey, California

1 9 6 3

NPS Archive

963

Griffith, W.

~~6845~~

PROGRAMMING
A REMOTE MONITOR AND DISPLAY SYSTEM
FOR A
TIME SHARING COMPUTER

by

- Webster Griffith
Lieutenant, United States Navy

This work is accepted as fulfilling
the thesis requirements for the degree of

MASTER OF SCIENCE

IN

ENGINEERING ELECTRONICS

from the

United States Naval Postgraduate School

ABSTRACT

The optimization of computer usage from both the users and the computer point of view is discussed. A system wherein several users simultaneously operate separate consoles is explained and specific requirements are laid down for a monitor and display system for a time sharing computer patterned to fit operations at the U. S. Naval Postgraduate School, Computer Center and particularly the CDC 1604 computer and the DD 65 display unit. A more stringent and less flexible system working on the same principle is described and programmed with explanations and instructions for expansion.

TABLE OF CONTENTS

Section	Title	Page
1.	Introduction	1
2.	System requirements	3
3.	Description of the general system	5
4.	Functions of the system	17
5.	Description of the example system	20
6.	Conclusion	30
7.	Illustrations	31-38
	Appendix-A MACHINE DD65A	39

LIST OF ILLUSTRATIONS

Figure	Page
1. Interrupt handler	31
2. Idle loop	32
3. DD65 keyboards	33
4. Flow chart of DD65A	34
5. Flow chart of KYBORD	35
6. Flow chart of MODE	36
7. Flow chart of PICGEN	37
8. Flow chart of PACK	38

1. Introduction.

The conflict between the large, fast and flexible digital computer and its slow human master has led to the loss of efficiency by having multi-million dollar computers idle as the human operators think.

One answer, and the subject of this thesis, is to let many users operate the computer simultaneously at separate and remote locations to combat the user inefficiency in the computer installation at the U. S. Naval Postgraduate School.

A control system which would permit the user to inspect, alter or compose a program from a remote console was desired. Available equipment included a CDC 1604 Computer¹ and a DD 65 Display Unit², the latter is a device with two cathode ray tubes, character generator, a random access memory for storage (reading is sequential to the character generator and associated logic circuits only), two keyboards for input and the necessary equipment to allow compatibility with the CDC 1604, CDC 160 and assorted radars. Direct linkage is not provided between the keyboards, the display and the user.

General specifications for this type of a system are difficult unless exact requirements are prescribed. The example used is therefore a limited case, however it is hoped that it can lead to a larger, fuller system.

¹Control Data Corporation, Computer Division, Control Data 1604 Computer Programming Manual, Publication 167, Undated.

²Instruction Manual for the DD 65 Display Unit.

The system should be capable of utilizing as much of the flexibility of the DD 65 as possible and still allow the user an easy to operate console.

Since the "Monitor" system in use at present is more or less adequate for normal stacked job operation, this system is aimed for the user requiring immediate access to his program or to its input or output. This requirement is such that little modification or addition is needed to expand the functions of the system to those mentioned in section 5.

The principle effort was directed to producing a system which would allow symbol manipulation in such a way as to achieve compatability between not only the equipment used but also the equipment and the users.

The example program is designed for use with the FORTRAN 60 Compiler¹ although the general system could handle as many compilers/assemblers as needed.

Besides the control system there are two principle subsystems:

- a. A keyboard to intermediate code compiler called KYBORD.
- b. A intermediate code to display unit compiler called PICGEN.

Other transcoders are of a more minor nature in design if not in importance.

¹Control Data Corporation, Computer Division, FORTRAN System for the Control Data 1604 Computer, Publication 087A, 1961.

2. System Requirements.

There will be two systems considered:

- a. The general system.
- b. The example system.

The example system is a lesser included part of the general system which is designed to demonstrate the practicability of the general system. The program of the example system and an explanation is given later. An "*" denotes requirements for the example system.

The system requirements are as follows:

- a. Computers
 - CDC 1604 and two CDC 160's.
 - *- CDC 1604.
- b. Consoles
 - CDC 1604, two CDC 160's and DD 65.
 - *- CDC 1604 and DD 65.
- c. Input/output (on-line)
 - 9 magnetic tape units, 3 flex punches, one of 3 radar's, experimental devices via a D/A converter, graph plotter, type-writer, DD 65 typewriter, and card reader.
- d. Compilers and Assemblers
 - FORTRAN 60, COOP MONITOR, JOVIAL, NELIAC, SCRAP, AR, and octial machine inputs.
 - *- FORTRAN 60.
- e. Libraries
 - Any proven programs in the above form.

- e. Libraries (Cont'd)* One integrated program as an example.
- f. Delays permitted - Only those delays normally encountered in FORTRAN 60 plus actual run times.
- g. Edit features - Insertions, deletions and exchanges can be made on programs on tape, in memory, and on the display tube.
 - *- No changes in tapes.
- h. Location of control
 - Control is in resident.
 - *- Control is in Program DD 65A.

3. Description of the general system.

a. Resident program.

Under present conditions at the U. S. Naval Postgraduate School Computer Center, the fastest input/output medium is binary magnetic tape. This is a most significant factor in the operation of the system control. .

A single resident and control program is required since the time delay in loading and unloading individual resident programs for individual consoles or special purpose usage would not permit the fulfillment of the requirement of instantaneous (for humans) response to DD 65 keyboard hits. This places a fundamental limitation on the system as a single resident program must be large for flexibility of control and display and small for flexibility of the input programs of the users.

The selection of the exact bias point is a delicate decision and the expeditious answer of providing all that might possibly be used or only as much as is presently used is not wise since, if the system is to be put in general use, the users must be told a firm number of memory cells which they have available. If the bias is altered the user's previously checked out programs may either not run due to insufficient room or not run efficiently due to not utilizing all the memory available.

The bias point was calculated as follows:

- | | | |
|-----|---------------------------|--------------------|
| (1) | Basic FORTRAN 60 resident | 5000 cells (octal) |
| (2) | DD 65 processor (KYBORD) | 2400 cells (octal) |

(3) Buffer for DD 65 memory. 1000 cells (octal)
(one buffer for each DD 65)

(4) Control system program 400 cells (octal)

Or 10,000 cells plus 1000 for each DD 65; over 1/8th of memory. Any further encroachment on the available memory space might easily limit the usefulness of the whole system and confine its use to only a portion of the user traffic.

b. *Simultaneity* and interrupt priority.

Since DD 65 keyboard hits must be processed without humanly discernible delay and since interrupting the keyboard in mid-routine requires the storage of many flag and switch settings and, also, since the keyboard routine requires less than one milli-second; interrupts actuated by a DD 65 not at present in control will be deferred if the console in control is processing a keyboard hit which does not directly activate a user program.

The interrupt priority list is as follows:

(1) A "normal" interrupt (e.g. "Interrupt on channel 5 active" or "Interrupt on Arithmetic Faults") will be processed as occurring.

(2) A DD 65 key hit interrupt can interrupt a running program but not another keyboard hit.

(3) A program run initiation can interrupt only when the control program is passive or idling. Once initiated a program can be interrupted by (b) or (a) above but not by another program which must wait until the present program is finished.

Priority Chart

	Idle	Program Being Run	Keyboard Hit Being Processed	Normal Int. In Process
Program to be run	Do	Wait	Wait	Wait
Key hit to be processed	Do	Do	Wait	Wait
Normal interrupt to be processed	Do	Do	Do	Do

Table 1

A flow chart illustrating the interrupt scheme is shown in Figures 1 and 2.

c. Keyboard decoder.

The keyboard decoder will analyze the input from the designated input typewriter. It will be noted that each DD 65 unit has two keyboards; Keyboard #1 has a modified and enlarged standard keyboard, Keyboard #2 is a special keyboard which can be programmed as desired and consists of 34 keys each with selectable lights, in addition the meaning of the keys can be altered by using any one of eight overlays which transmits a three bit code along with the six bit code of the individual key. Even with the large number of keyboard #2 keys available, it is still inconvenient to govern all of the possible situations from Keyboard #2 only. Therefore, I have made Keyboard #2 the category selector and have let Keyboard #1 select subcategories as appropriate. For example, if "MODE" is selected by Keyboard #2, then "CHARACTER", "VECTOR", or "SPOT" is selected by hitting "C", "V", or "S" respectively on Keyboard #1.

In general then, the KEYBOARD DECODER analyzes the key hit in light of the key hit history (e.g. A "C" hit after "MODE" has been hit has an entirely different meaning than a "C" hit after "TYPE" has been hit) and sets switches and selects subroutines which will be used later in the KEYBOARD PROCESS portion of the program. The system of category selection on Keyboard #2 and subcategory selection on Keyboard #1 was chosen because:

(1) The number of Keyboard #2 keys under one overlay is limited to 34. Even the limited example system has 48 subcategories plus numerical selects.

(2) If multiple Keyboard #2 hits were used to select subcategories, a problem as to labeling would arise. One can label the category key easily but the subcategory code could not be labeled and would require the user to either remember or make continued reference to a code sheet.

(3) By allowing Keyboard #1 to denote subcategory selections, easily remembered codes can be used (e.g. "S" for small, "R" for right, "P" for pipper, etc.). In addition the number section and sign keys are handy (e.g. a right margin 63 spaces to the right of center is selected as follows: hit "RIGHT MARGIN" key on Keyboard #2 and then hit "+", "6", and "3").

d. Keyboard processor.

Once the KEYBOARD DECODER has completed its operations the KEYBOARD PROCESSOR is called and carries out the execution of the steps required by the particular hits.

e. The intermediate code.

Since information in the DD 65 memory is not retrievable except to be displayed on the DD 65 scopes, some additional storage is required. With our speed requirements only CDC core will serve. This information could be in an of three forms:

(1) Raw input as received in the computer.

(2) Processed input in a form directly transferable to the DD 65 memory.

(3) Processed input of a uniform nature but not necessarily in the form suitable for DD 65 memory.

The first form requires that every type of input be reprocessed by its own peculiar coder/decoder whenever updating or retrieval are required. The second form requires multiple input coders but only one decoder, however the decoder must unpack bits in highly condensed data words and would be slower than the third form which uses a special coder to place each type of input into a simple intermediate code. This code is stored in a buffer area and is identical for all inputs. This is the system used. Each form of input (e.g. DD 65 typewriter, BCD magnetic tape, flex tape) has its own coder which stores the transformed input in an area called PICBUF. Each of these coders are reciprocal and one can go from flex tape to PICBUF and back without loss. However, since all of the input forms differ in some respect, going from the DD 65 typewriter to PICBUF, to magnetic tape, back to PICBUF, and finally to the DD 65 display could, but would not necessarily, involve some loss.

f. Picture generator.

The picture generator (PICGEN) transforms the intermediate coded PICBUF into the form suitable for the DD 65 and transmits this code to the DD 65. There are several ways to approach the use of PICGEN for updating. One, and normally the fastest, is to only process or reprocess the last word sent to the DD 65 memory. This method has the disadvantage of being difficult to edit and some forms of editing are particularly cumbersome. Another method is to update only the last line of display on the DD 65. This allows easy editing of the last line but has the same difficulty encountered in the first method in deeper editing. The method chosen is to update the entire DD 65 memory each time a character is added to PICBUF. This is by far the slowest method but does allow easy and complete editing features. Since PICGEN returns to the control program as soon as an "end of file" symbol is reached, only when PICBUF is nearly full will the time required to decode be of any consequence. The time from entry to exit under worst case conditions is .36 seconds and under normal conditions would be about .10 seconds. For example, a man typing at a rate of 50 words a minute averages .20 seconds per key hit, therefore only a fast typist typing 128 characters per line at the bottom of the page would notice any reduction in his normal typing speed.

THE PICGEN CODE

- 00 - Begin string. All bytes after this are transmitted directly to the DD 65 until a 77 byte is used.
- 01 - Disregard all bytes until this one is processed.
- 10 - Set for character mode.
- 11 - Set for vector mode.
- 12 - Set for spot mode. The next three bytes represent the X (1st 9 bits) and Y (last 9 bits) location of a ".".
Exit from the string is automatic. If more than one spot is to be sent, each must be preceded by a begin string code (00).
- 20 - Set X location equal to the value of the next byte.
- 21 - Set Y location equal to the value of the next byte.
- 22 - Set X location equal to - the value of the next byte.
- 23 - Set Y location equal to - the value of the next byte.
- 30 - Set for small size characters.
- 31 - Set for medium size characters.
- 32 - Set for large size characters.
- 40 - Set for display on right tube.
- 41 - Set for display on left tube.
- 50 - Set for normal intensity.
- 51 - Set for bright intensity.
- 60 - Set for auto-incrementation right.
- 61 - Set for auto-incrementation down.
- 76 - End of file, exit PICGEN.
- 77 - Disregard this byte.

THE CHARACTER CODE

00 - Blank	26 - W.	54 - \$.
01 - 1.	27 - X.	55 - ↑.
02 - 2.	30 - Y.	56 - .
03 - 3.	31 - Z.	57 - ^.
04 - 4.	32 -].	60 - +.
05 - 5.	33 - ,.	61 - A.
06 - 6.	34 - (.	62 - B.
07 - 7.	35 - →.	63 - C.
10 - 8.	36 - Tab.	64 - D.
11 - 9.	37 - ≥.	65 - E.
12 - ∅	40 - _.	66 - F.
13 - =	41 - J.	67 - G.
14 - 7.	42 - K.	70 - H.
15 - ::	43 - L.	71 - I.
16 - '.	44 - M.	72 - <.
17 - [.	45 - N.	73 - ..
20 - Space	46 - O.	74 -).
21 - /.	47 - P.	75 - {.
22 - S.	50 - Q.	76 - CR.
23 - T.	51 - R.	77 - End of string.
24 - U.	52 - }.	
25 - V.	53 - \$.	

THE VECTOR CODE

The vector code describes the size, direction and "on" condition of the vector in six bits as follows:

0X - Blank, small vector.

1X - Blank, large vector.

4X - Unblank, small vector.

5X - Unblank, large vector.

X0 - To the right.

X1 - To the right and up (elevated 45° to the right).

X2 - Up.

X3 - To the left and up (elevated 45° to the left).

X4 - To the left.

X5 - To the left and down (depressed 45° to the left).

X6 - Down.

X7 - To the right and down (depressed 45° to the right).

77 - End of vector string.

THE SPOT CODE

The spot code consists of 18 bits, 3 bytes or 6 octal numbers. The first 9 bits represents the X location and the last 9 represents the Y location. The tube is divided into 512 (1000 octal) lengths in each direction with the (0,0) location at the center. The location code is in one's complement notation.

Examples: (1) 377000 is right center

(2) 400400 is the lower left corner.

(3) 000000, 777777, 000777, and 777000 are all dead center.

An example: If the first few words in PICBUF looked like this:

10 10 31 40 50 60 23 77

21 77 00 62 65 67 71 45

20 70 65 51 65 77 77 76

It would mean to start the file (01), use character mode (10) of medium size (31) on the right tube (40) of normal intensity (50) and increment to the right (60). Start at the left (23 77) upper (21 77) corner. String follows (00). "BEGIN HERE". Stop the string (77) and exit (76).

g. Other decoder or transcoders.

Other media must be translatable to and from the PICBUF code. In this case; Flex paper tape, binary magnetic tape, BCD magnetic tape, binary core and BCD core. The actual source of the code is not important to the code since all will be delivered to a buffer area in core and separate routines will load and dump into these buffers. Therefore, we need three decoders and their reciprocals; Flex, BCD, and binary to PICBUF code and PICBUF code to Flex, BCD, and binary in addition of course, DD 65 keyboard to PICBUF and PICBUF to DD 65 display.

(1) Flex to PICBUF.

(a) Upper case to medium size and lower case to small size characters. The reverse in numerals.

(b) Change color of ribbon works a toggle switch on the intensity code.

(c) Characters not present to "\$".

(d) "&" to plus sign.

(e) "*" to times sign.

(2) PICBUF to Flex.

The reverse of (1) above except large to upper case and all brackets to "(" and ")". It might be pointed out here that since PICBUF code is the most flexible, care must be taken in forcing some transcriptions like "Increment Down". One can force other media to increment down, however, one can not make them backspace. Impossible transcriptions like "Left Tube" are ignored.

(3) BCD to and from PICBUF

This is straight forward since the FORTRAN 60 BCD characters are relatively few and non-character selects absent.

(4) Binary to and from PICBUF.

The octal, 16 number representation is used.

(5) Vector to and from Graph Plot.

This transcription in either direction is within the capabilities of the system but has not been investigated further.

h. A call to run a user's program.

If the size and complexity of the system are to be retained within bounds, and automatic super-master program which could bootstrap, control time length of runs under various priority conditions, permit check out program runs, interpretively run non-checked out programs, etc.; can not

be included nor do I feel it would be desirable. Since four users are being kept busy, the expense of one operator would be well worthwhile especially since monitor programs could be done whenever the computer was between runs. Therefore, all programs will be allowed to run if space permits. If a bad program clobbers the system the operator can easily bootstrap and he can cause premature exits from programs which take too long. Here, it might be pointed out is another advantage of the human operator since the users through the intercom can make special requirements known to the operator who can allow or not extra long programs and advise the other users if long delays are expected. The operator could also load special libraries and activate on-line equipment if desired.

4. Functions of the system.

Once the control system is in operation, it can be used for several functions; the diversity of which is determined by the sophistication of the program and the equipment available

a. Student's computer.

This mode of operation is designed for use by one not familiar with all intricacies of computer operation. In fact only previously checked out programs on a specific library will be permitted to run. The only option of the user will be to pick up the desired program and insert data. In most cases the output will be to the DD 65 display with an option for a permanent record when desired. Suitable error exits must be made for input data which is not appropriate to the program and likewise all output data must be tested for its fit on the display and printer (or graph plot).

Consequently only programs of a general and continuing nature would warrant the programming of a problem. As an example an instructor could ask a class studying Fourier analysis to input wave forms and observe the output. The student in this case would place vectors on the display to represent the wave form and a bar graph representation could represent the amplitude of the sin and cos coefficients.

In addition, if the results of a program under many varying parameters is to be studied at great length, this mode could allow untrained observers to procure data with little effort.

b. Check-out computer.

This mode of operation would allow the user to write, edit, compile and run FORTRAN programs.

One often is faced with a long time delay from the completion of coding to the commencement of useful runs due to the time consumed debugging their programs. In the check-out computer, a program to be tested is loaded on a tape unit or fed from the satellite 160. The user then requests from the DD 65 a compilation of the program. If compilation errors result they will appear on the display. The user calls in his program to the display tube, makes the necessary corrections, and reloads the tape unit. The process is repeated until all errors of compilation are eliminated. Then the program is run and all errors in running are likewise corrected. When the program is completely error free the user can ask for an on or off-line punch of new cards containing the revised program.

Previously a user had to either wait for the next monitor run or; if operating the computer personally, the computer was in an idle state for long periods while cards were changed, etc.

This computer is not as simple to use as the students' but would require little more knowledge than is needed for FORTRAN programing.

This computer would also be used for parameter variation problems as in the student's computer but would not

require the rigorous testing and "libing" of the latter.

c. Experimental display computer.

Using this mode, on-line experimental data is fed through the satellite CDC 160 to the CDC 1604 and the processed information is displayed on the DD 65.

This offers a unique opportunity to adjust not only the experimental equipment but also the processor parameters in a rapid sequence which would allow much quicker optimization of the equipment.

d. Teaching machine.

Although the student's computer is a teaching machine some such devices are of such a special nature as to require individual libraries and would have to be programmed as needed.

e. Learning machine.

As in d. above, special programs would be required.

5. Description of the example program.

As previously mentioned, this program is a lesser included program of the general system designed to show the latter's practicability by providing a working system and to form a basis upon which to build the general system.

a. Basic program (DD 65A).

This program is the executive or control program and determines when, what and how soon a subroutine is to be called and what the inputs to the subroutine will be. The Idle Loop and Interrupt Handler would be inserted here in the general system but they are not necessary as such in this limited system. Note that the status reports are sent to DD 65A via the Q-register.

b. Keyboard handler (KYBORD).

This subroutine incorporates both the Keyboard Decoder and the Keyboard Processor of the general system. It has two arguments: 1. The input byte, 6 bits, right justified in the A-register, 2. The keyboard from which it came in the Q-register. The output, also in the Q-register, indicates where the next keyboard hit should originate, however, this is of information value only to the DD 65A since the latter can over-ride any recommendations of KYBORD.

In this and other subroutines routing is used whereby information in one hit alters the processing route of a future hit(s).

The keys are grouped as follows:

- (1) Keys which set the characteristics of the

display when Keyboard #1 is subsequently hit.

(a) "Mode"

When the "Mode" key (see figure 6) is hit the routing is adjusted for the following possibilities:

1. "C" is hit next.
2. "V" is hit next.
3. "S" is hit next.
4. A Keyboard #1 key not "C", "V", or "S" is hit next.

5. A Keyboard #2 key is hit next.

If case 5. action is taken in accordance with the new Keyboard #2 hit and the "MODE" hit is ignored. In case 4. an error exit results which displays a flashing "KBI ERROR" on the typing tube and the control program goes into idle. In cases 1., 2. and 3. "CHARACTER", "VECTOR", or "SPOT" is displayed on the status tube and appropriate initiating action is taken in the PICBUF region. At the completion of the action DD 65 awaits the next Keyboard #2 hit.

(b) "SIZE"

The remainder of the characteristic Keyboard #2 keys are processed in a similar fashion to "MODE" and inappropriate hits are handled exactly the same.

Appropriate Keyboard #1 hits are:

1. "S" for "SMALL 126"
2. "M" for "MEDIUM 63"
3. "L" for "LARGE 31"

The indications of size and maximum number of characters per line are displayed on the status tube. In addition and in conjunction with the selection for the margins the actual number of characters per line ("CHAR/LINE XXX") is calculated and displayed on the status tube.

(c) "TUBE"

Appropriate Keyboard #1 hits are:

1. "R" for "RIGHT TUBE".
2. "L" for "LEFT TUBE".

It is well to mention that all of the characteristics selection displays are stored in a separate section from PICBUF and therefore are not included in any dump or decoding of PICBUF.

(d) "INT" (intensity)

Appropriate Keyboard #1 hits are:

1. "N" for "NORMAL".
2. "B" for "BRIGHT".

(e) "INCR" (increment).

1. "R" for "INCR RIGHT".
2. "D" for "INCR DOWN".

(f) "SPACING"

Appropriate Keyboard #1 hits are:

1. "S" for "SINGLE" spacing.
2. "D" for "DOUBLE" spacing.
3. "T" for "TRIPLE" spacing.

(g) "PIP"

The "PIP" key is an on/off switch. When

on the "PIP" key is lighted an "X" appears on the typing tube in the position of the pip. The pip's position is controlled by the four keys "UP", "RIGHT", "DOWN" and "LEFT" which are called direction keys and operate as follows: If the direction key hit is not the same as the last direction key hit, the pip is not moved, but if the direction key is the same the pip is moved in the direction indicated by an amount equal to one space more than it moved the last time. For example if the "UP" key was hit and then the "RIGHT" key was hit ten times in succession, the pip would move as follows:

"RIGHT" hits

	1st	2nd	3rd	4th	5th	6th	7th	8th	9th	10th
Spaces moved to the right	0	1	2	3	4	5	6	7	8	9
Relative pip position	0	1	3	6	10	15	21	28	36	45

The pip has end around action, that is if one continues to hit the "RIGHT" key after the pip is on the right extreme of the tube, the pip will appear on the extreme left of the tube.

(h) "LEFT MARGIN".

Appropriate Keyboard #1 hits.

1. "P" for "PIP" where the left margin is taken from the horizontal position of the pip.

2. "+" or a "-" followed by two octal numbers where the left margin is taken from the number given considering the tube center to be 00 and to extend to

77 (oct) spaces in either direction. An indication of the margins is displayed on the typing tube by an upward pointing arrow and the octal position of the arrow displayed underneath.

(i) "RIGHT MARGIN" (same as left margin).

(j) "TAB SET".

This hit does not wait for a Keyboard #1 hit but uses the horizontal position of the pip as the desired tab setting. A tab is displayed as a caret on the typing tube (top). There are 16 tabs available and until all 16 are used elsewhere a tab will always be set at the right extreme of the tube.

(k) "TAB CLEAR"

The pip is again used as a position reference. If no tab is in the position indicated, the "TAB CLEAR" key will blank thrice and the program will return to idle.

(l) "MARKER"

The marker indicates where the next character will be placed or where the end of the last vector is (an aid in using blank vectors) as appropriate. The particular symbol to be used as a marker is determined by the Keyboard #1 key (including the space bar) immediately after the "MARKER" key is hit.

(m) "START"

The "START" key initially locates the marker and operates as follows:

1. "T" - Start at the top of the tube and at the left margin.

2. "P" - Start at the position indicated by the pip.

3. "Y" - Start at the left margin and vertically as indicated by the pip.

(n) "TRACK BALL"

The "TRACK BALL" key is an alternate way to position the PIP.

(o) General

All of the preceeding keys affect the character mode but in the vector mode all indications are ignored except Mode, Tube, Marker and Start and in the spot only Mode, Tube, Pip, and Start are recognized. However, one may switch from mode to mode without reselecting. For example if Size is set for Large and one then transfers to Vector mode using small vectors, a transfer back to character Mode will result in using Large size characters unless altered.

The light associated with a key will remain on as long as additional information is required from Keyboard #1. A blinking light indicates that the function of the key could not be completed and when the blinking stops control is returned to the executive program.

(2) Set to type

The "TYPE" key activates Keyboard #1 and until another Keyboard #2 key is hit the Keyboard #1 keys will

operate as previously selected.

(a) In the "CHARACTER" mode normal typing procedures are used and, in addition, an automatic carriage return will result when the last character or space which will fit between the margins has been typed. The caret of the marker will always indicate where the next character will go. Tabs operate in a normal fashion except tabbing past the left margin will result in an automatic carriage return.

(b) In the "VECTOR" mode only part of the keyboard is activated, that is the numerals 0 through 9 and the letters "S", "M", "L", and "E". The letters and "0" and "5" do not cause a vector to be formed but modify subsequent vectors as follows:

1. "S" - A small vector.
2. "M" - A large vector.
3. "L" - Two large vectors.
4. "E" - Four large vectors.
5. "5" - Unblank vectors.
6. "0" - Blank vectors.

The remaining numerals from vectors and their direction can be determined by the number key's relative position from the "5" key. They are as follows:

1. "1" - Southwest (The top of the tube is North)
2. "2" - South
3. "3" - Southeast

4. "4" - West
5. "6" - East
6. "7" - Northwest
7. "8" - North
8. "9" - Northeast

(c) In the "SPOT" mode Keyboard #1 is not activated. While in this mode, a spot is generated each time the "TYPE" key is hit and the position of the spot is taken from the pip.

(2) Edit features

(a) The Keyboard #1 "blank" key acts as a combination backspace and delete.

(b) "CHAR EDIT" locates the character indicated by the pip and will then proceed as follows:

1. If indicated character can not be found the "CHAR EDIT" key will blink and exit the routine.

2. If "D" on Keyboard #1 is hit, the indicated character is deleted and all characters to the right on that line are moved to the left one space.

3. If "R" on Keyboard #1 is hit, the indicated character is deleted and replaced by the next hit on Keyboard #1.

4. If "I" on Keyboard #1 is hit, the next hit on Keyboard #1 is inserted just to the left of the pip. This key operates only in "CHARACTER" mode.

(c) "LINE EDIT" locates the line and character indicated by the pip and proceeds as follows:

1. If the line and character indicated can not be found, the "LINE EDIT" key will blink and exit the routine.

2. If "D" is hit the entire line indicated will be deleted and the following lines moved up to fill in.

3. If "R" is hit the indicated character and all to the right on the indicated line will be deleted. Keyboard #1 hits will then be inserted in sequence starting with the location of the pip and continue until either the end of the line or until "CR" is hit at which time the routine will exit. "TAB" will be ignored.

(d) "RESET" will extinguish all Keyboard #2 lights except its own and end the current string.

(e) "CLEAR" clears PICBUF and prepares it for a refill (starts file). It also clears the displays which are no longer appropriate.

c. Picture generator (PICGEN)

PICGEN is a transcoder which converts the intermediate PICGEN code to the machine code acceptable to the DD 65 display. The PICGEN code is composed of two parts; one of which, the string code, is merely a modified BCD code for transmitting specific characters, vectors or spots. The other code establishes the characteristics of the display. Both codes are in six bit bytes, 8 bytes to a word and words in numerical increasing order starting at address PICBUF+1. The program PICGEN shuffles through PICBUF byte by byte in sequence forming the proper skeletal words for input to the

DD 65 display until the "00" code causes a shift in mode to string processing whence each byte is inserted in the proper location word by word until the "77" code is reached which terminates the string and the original mode of operation is reactivated.

Since all 64 codes in the string code are meaningful there are no error exits in that mode, however, impossible characteristic codes cause an exit and error display on the right tube.

The process is continued until a "76" code is processed at which time an exit from PICGEN is taken. If a "76" code is not processed when PICBUF is exhausted an exit will also occur.

d. Character locator (CHALOC)

CHALOC uses the pip location to find a specific character in PICBUF and relay the information back to the calling program. If no character is found at the designated location, a -1 is inserted in the A-register. This routine could be expanded to trace vectors and spots but at present this cannot be handled nor can the vertical incrementation be processed.

6. Conclusion.

A time sharing, remote monitor and display system is feasible with equipment presently available at the U. S. Naval Postgraduate School.

The use of a DD 65 Display Unit in conjunction with a CDC 1604 computer with a FORTRAN 60 compiler permits a user to monitor and view his program and its input and output with greater ease and at the same time not interfere or delay appreciably with the normal operations on the main console.

An intermediate code or language of some sort is necessary to allow compatability of various input and output equipment. Its form will be determined by the specific use to be made of the system and the equipment peculiarities, however, the more general code will allow easier additions and modifications.

INTERRUPT HANDLER

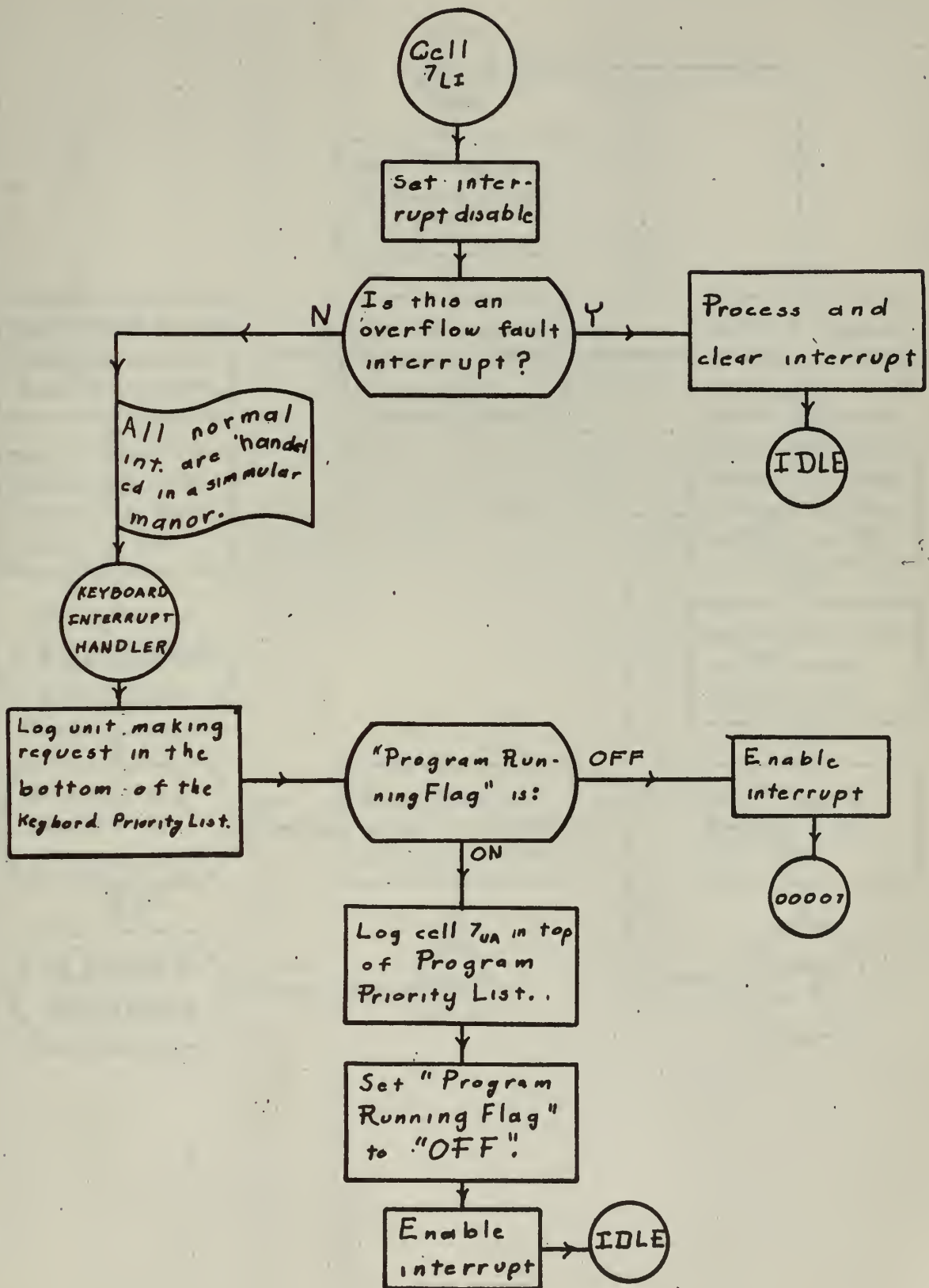


Fig 1

IDLE LOOP

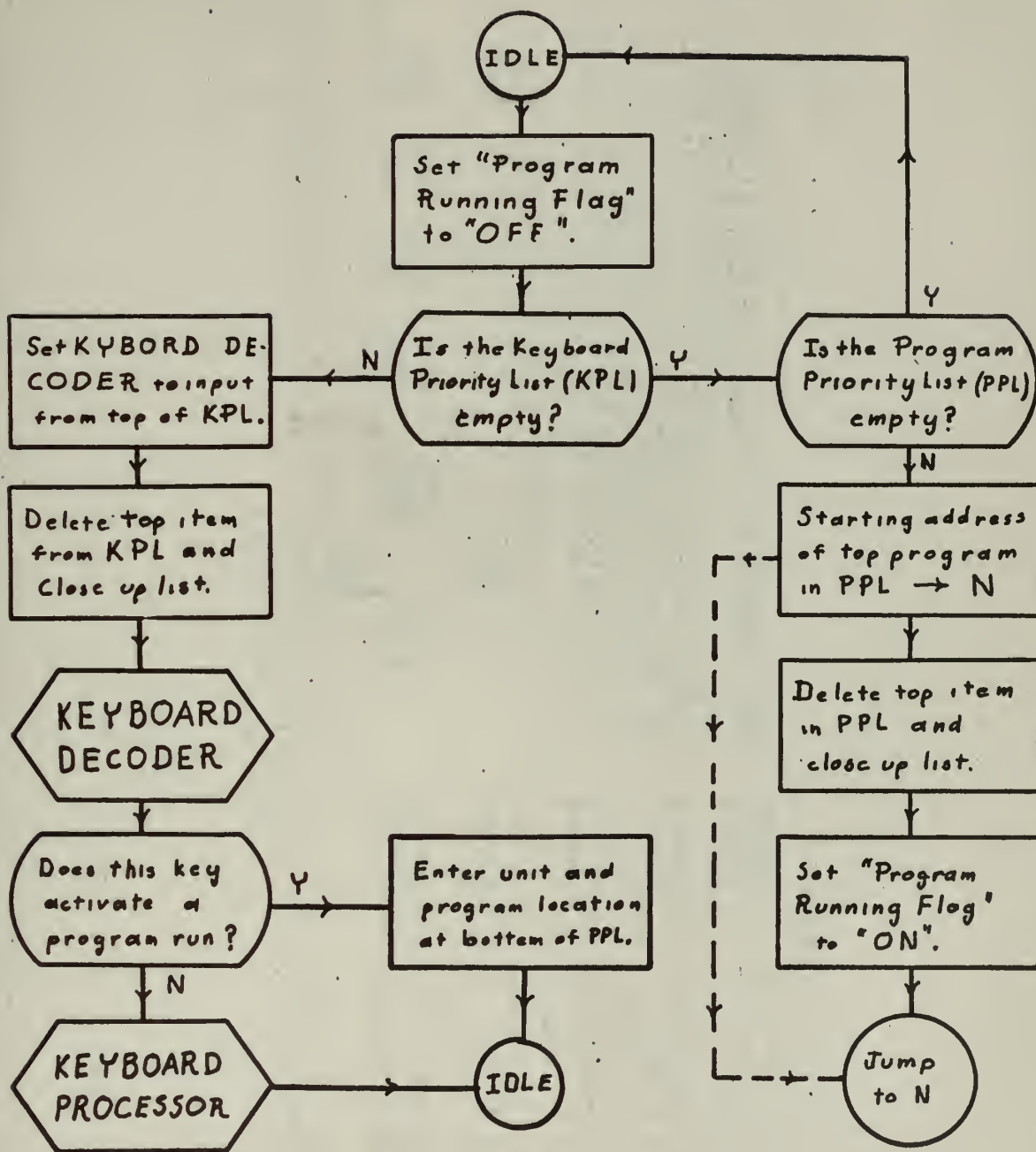
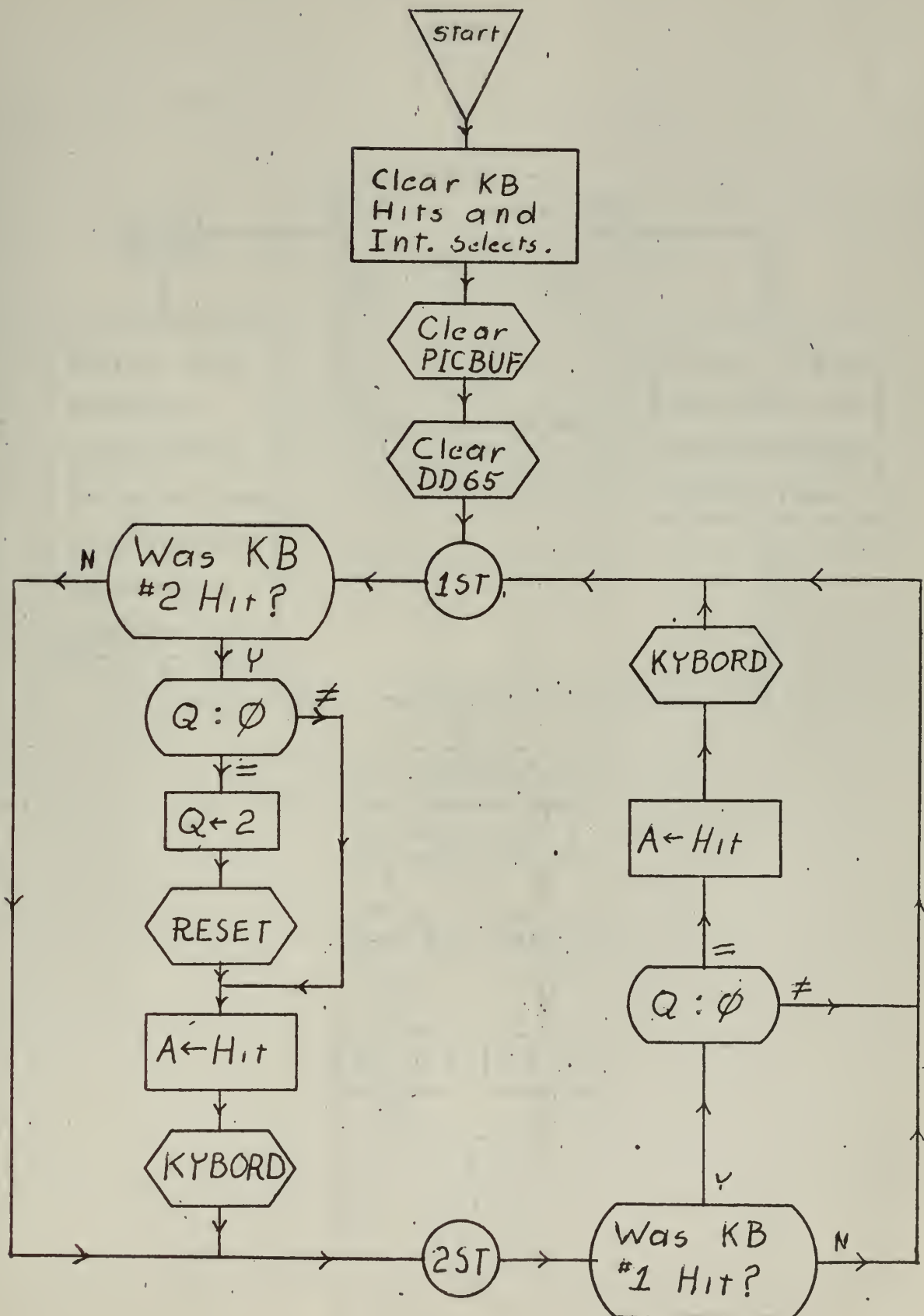


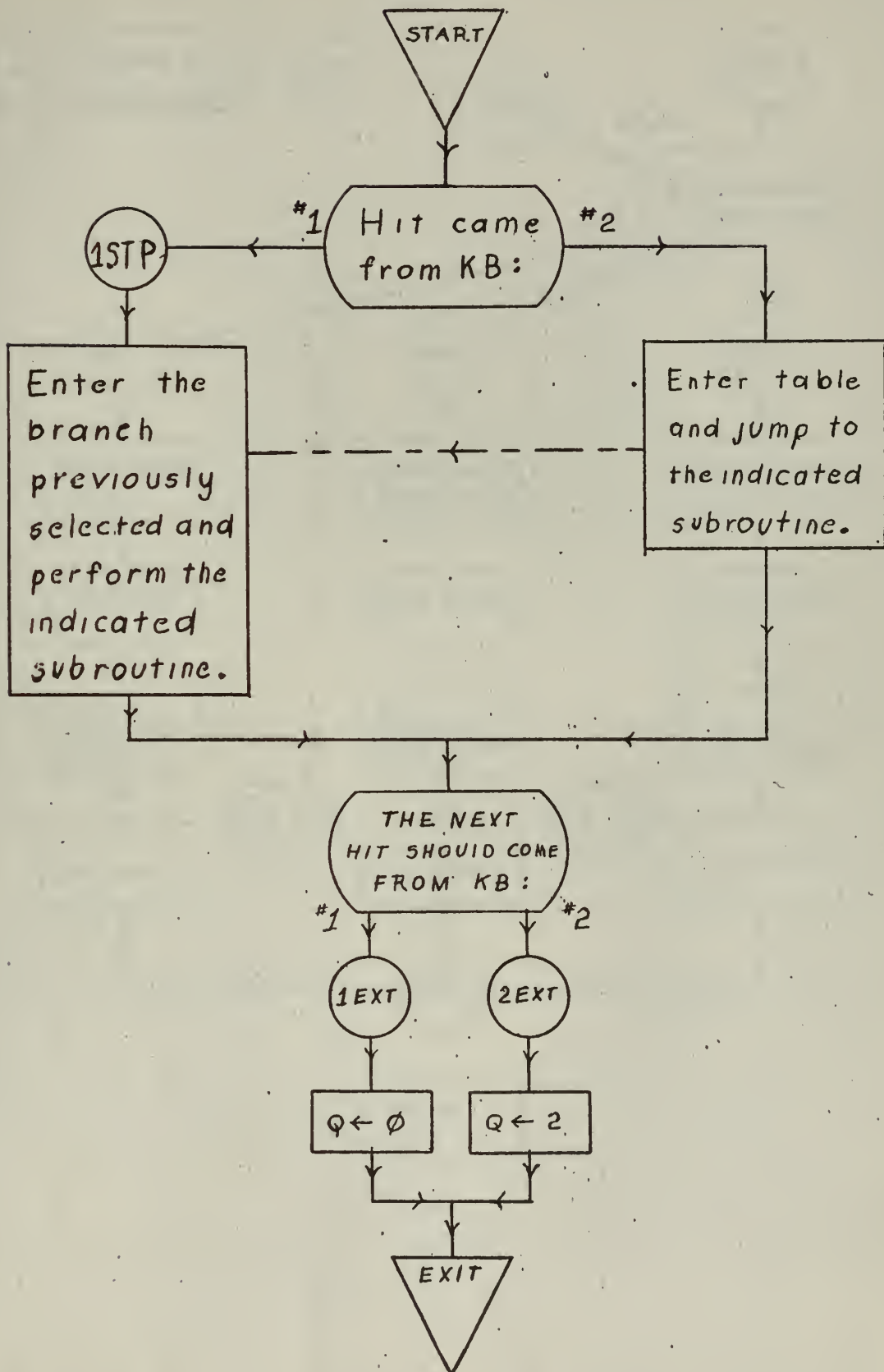
Fig 2

[illegible]

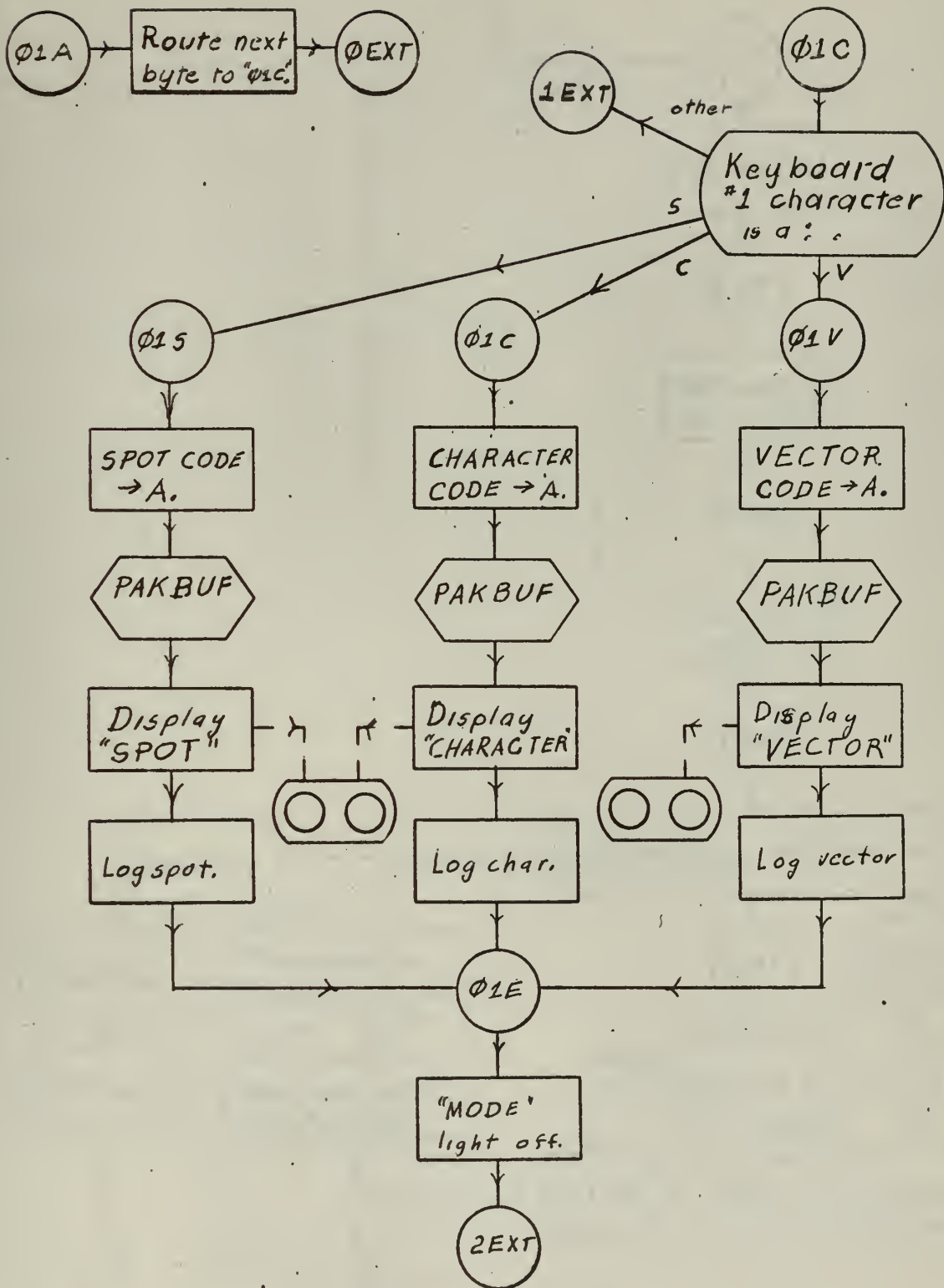
DD 65 A



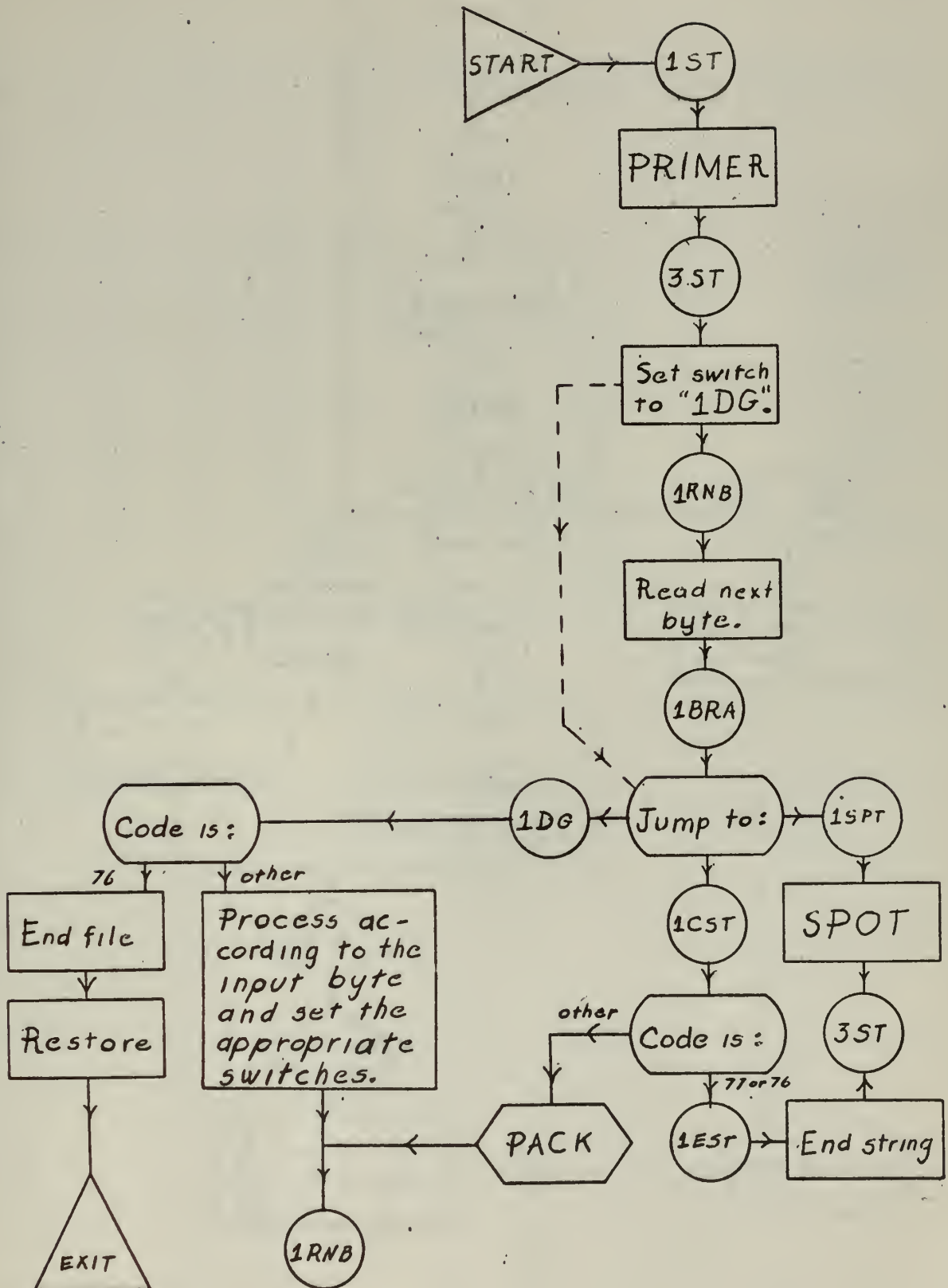
KYBORD



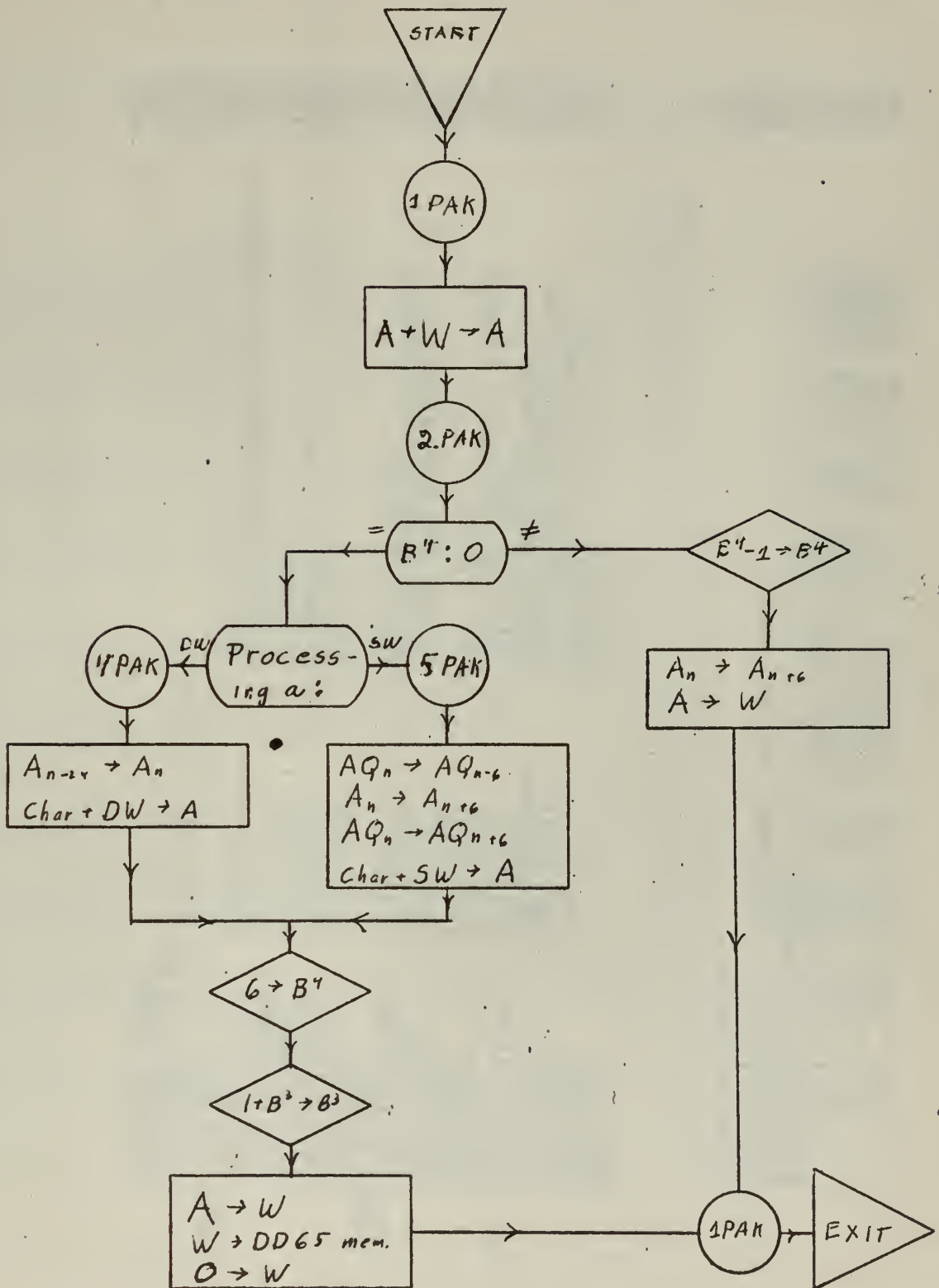
MODE



PICGEN



PACK



1 TBL	INA(1 TBL)	SAU(L+1)	•	ENTER TABLE.	MAK
	SLJ(N)	SLJ(01A)	•	LIGHT MODE.	MAK
	EXF(77202B)	SLJ(02A)	•	SIZE.	MAK
	EXF(77204B)	SLJ(03A)	•	TUBE.	MAK
	EXF(77210B)	SLJ(04A)	•	INTENSITY.	MAK
	EXF(77220B)	SLJ(05A)	•	INCREMENT.	MAK
	EXF(77240B)	SLJ(11A)	•	LEFT MARGIN.	MAK
	EXF(77302B)	SLJ(12A)	•	RIGHT MARGIN.	MAK
	EXF(77304B)	SLJ(13A)	•	TAB SET.	MAK
	EXF(77310B)	SLJ(14A)	•	TAB RESET.	MAK
	EXF(77340B)	SLJ(15A)	•	SPACING.	MAK
	EXF(77402B)	SLJ(2EXT)	•		MAK
	EXF(77404B)	SLJ(2EXT)	•		MAK
	EXF(77410B)	SLJ(2EXT)	•		MAK
	EXF(77420B)	SLJ(2EXT)	•		MAK
	EXF(77440B)	SLJ(31A)	•	TYPE.	MAK
	EXF(77502B)	SLJ(32A)	•	START.	MAK
	EXF(77504B)	SLJ(33A)	•	MARKER.	MAK
	EXF(77510B)	SLJ(34A)	•	CLEAR.	MAK
	EXF(77520B)	SLJ(35A)	•	TEST LIGHTS.	MAK
	EXF(77540B)	SLJ(41A)	•	PIP.	MAK
	EXF(77602B)		•	TRACK BALL.	MAK
	SLJ(42A)		•		MAK
	EXF(77610B)	SLJ(2EXT)	•	CHARACTER EDIT.	MAK
	EXF(77620B)	SLJ(44A)	•	RESTART MACHINE DD65A.	MAK
	EXF(77640B)	SLJ(2EXT)	•		MAK
	EXF(77702B)	SLJ4(RST)	•		MAK
	EXF(77704B)	SLJ(2EXT)	•	LINE EDIT.	MAK
	EXF(77710B)	SLJ(2EXT)	•	RESET.	MAK
	EXF(77720B)	SLJ(54A)	•	UP.	MAK
	EXF(77740B)	SLJ(55A)	•	RIGHT.	MAK
	SLJ(61A)		•	DOWN.	MAK
	SLJ(62A)		•	LEFT.	MAK
	SLJ(63A)		•		MAK
	SLJ(64A)		•		MAK
* EXIT	ENQ(0)	SLJ(1ST)	•	SET FLAG10 = 0 (KBI).	MAK
OEXT	ENI2(2)	EXF7(77010B)	•	KEYBOARD 1 ERROR EXIT.	MAK
TEXT	EXF(77010B)	OUT3(ERR2)	•	SEL AND WAIT FOR DD65 MEM.	MAK
	ENI3(2)	SLJ4(3WAT)	•		MAK
		EXF7(77010B)	•	SEL AND WAIT FOR DD65 MEM.	MAK
	EXF(77010B)	SLJ4(3WAT)	•	SEL AND WAIT FOR DD65 MEM.	MAK
	OUT(ERR1B)	EXF7(77010B)	•	SEL AND WAIT FOR DD65 MEM.	MAK
	EXF(77010B)	SLJ4(3WAT)	•		MAK
	OUT(ERR1C)	EXF7(77010B)	•		MAK
	EXF(77010B)	OUT3(ERC2)	•		MAK
	ENI3(2)	SLJ4(3WAT)	•		MAK

2EXT	IJP2(1EXT+1)	SLJ(0EXT)	• SET FLAG10 = 2 (KB2).	MAK
* MODE	ENQ(2)	SLJ(1ST)	• S=SPOT.	MAK
01A	-C=CHARACTER	, V= VECTOR,	• ROUTE NEXT HIT.	01K
01AE	ENA(01C)	SLJ(0EXT)	• NO PICGEN.	01K
01C	SAU(1STP)	INA(-63B)	• CHARACTER.	01K
	ENI6(1)	INA(36B)	• VECTOR.	01K
	AJP(01CH)	INA(3)	• SPOT.	01K
	AJP(01V)	SLJ(1EXT)	• SEL AND WAIT FOR DD65 MEM.	01K
01CH	AJP(01S)	SLJ4(1PB)	• SEL AND WAIT FOR DD65 MEM.	01K
	ENA(10B)	EXF7(77010B)	• SEL AND WAIT FOR DD65 MEM.	01K
	EXF(77010B)	OUT(CHA1)	• MODE = 0-CA, 1-VCT, 2-SPOT.	01K
01E	OUT(CHA2)	EXF7(77010B)	• EXIT MODE.	01K
	STA(MODE)	ENA(0)	•	01K
	SLJ(2EXT)	EXF(77203B)	•	01K
01V	ENA(11B)	SLJ4(1PB)	• SEL AND WAIT FOR DD65 MEM.	01K
	EXF(77010B)	EXF7(77010B)	• SEL AND WAIT FOR DD65 MEM.	01K
	EXF(77010B)	OUT(VCT1)	•	01K
	EXF(77010B)	OUT(VCT2)	•	01K
	ENA(1)	SLJ(01E)	•	01K
01S	ENA(12B)	SLJ4(1PB)	• SEL AND WAIT FOR DD65 MEM.	01K
	EXF(77010B)	EXF7(77010B)	•	01K
	EXF(77010B)	OUT(SPT1)	• SEL AND WAIT FOR DD65 MEM.	01K
	EXF(77010B)	EXF7(77010B)	•	01K
	ENA(2)	OUT(SPT2)	•	01K
* SIZE	-S=SMALL, M=MEDIUM, L=LARGE.	SLJ(01E)	• ROUTE NEXT HIT.	01K
02A	ENA(02C)	SLJ(01AE)	• SMALL.	02K
02C	ENI6(1)	INA(-22B)	• MEDIUM.	02K
	AJP(02S)	INA(-22B)	• LARGE.	02K
	AJP(02M)	INA(1)	• SIZE = 4-SMALL, 8-MED, 16-LARGE.	02K
02S	AJP(02L)	SLJ(1EXT)	• SEL AND WAIT FOR DD65 MEM.	02K
	ENA(30B)	SLJ4(1PB)	• SEL AND WAIT FOR DD65 MEM.	02K
	EXF(77010B)	EXF7(77010B)	• CAL SIZE / 2 = SIZE2.	02K
	EXF(77010B)	OUT(SML1)	• RECAL SPACING.	02K
	EXF(77010B)	EXF7(77010B)	• RECAL CHAR/LINE.	02K
	EXF(77010B)	ENA(4)	• EXIT.	02K
02E	OUT(SML2)	ARS(1)	• SEL AND WAIT FOR DD65 MEM.	02K
	STA(SIZE)	SLJ4(15Z)	•	02K
	STA(SIZE2)	SLJ4(1MAR)	•	02K
	EXF(77205B)	SLJ(2EXT)	•	02K
02M	ENA(31B)	SLJ4(1PB)	• SEL AND WAIT FOR DD65 MEM.	02K
	EXF(77010B)	EXF7(77010B)	•	02K
	EXF(77010B)	OUT(MED1)	• SEL AND WAIT FOR DD65 MEM.	02K
	EXF(77010B)	EXF7(77010B)	•	02K
	ENA(10B)	OUT(MED2)	•	02K
		SLJ(02E)	•	02K

02L	ENA(32B)	SLJ4(1PB)	•	SEL AND WAIT FOR DD65 MEM.	02K
	EXF(77010B)	EXF7(77010B)	•		02K
		OUT(LRG1)	•		02K
	EXF(77010B)	EXF7(77010B)	•	SEL AND WAIT FOR DD65 MEM.	02K
		OUT(LRG2)	•		02K
	ENA(20B)	SLJ(02E)	•		03K
* TUBE - R=RIGHT, L=LEFT.			•	ROUTE NEXT HIT.	03K
03A	ENA(03C)	SLJ(01AE)	•	RIGHT.	03K
03C	ENI6(1)	INA(-51B)	•	LEFT.	03K
	AJP(03R)	INA(6)	•	NOTA.	03K
	AJP(03L)	SLJ(1EXT)	•		03K
03R	ENA(40B)	SLJ4(1PB)	•	PLACE INDICATORS ON LEFT TUBE.	03K
	ENI1(35B)		•		03K
	LDA1(CHAI)	SCL(MK11)	•		03K
	STAI(CHAI)	IJP1(L-1)	•		03K
	ENI1(27B)		•		03K
	LDA1(RM2)	SST(MK11)	•		03K
	STAI(RM2)	IJP1(L-1)	•	SEL AND WAIT FOR DD65 MEM.	03K
	EXF(77010B)	EXF7(77010B)	•		03K
		OUT(RHT1)	•		03K
	EXF(77010B)	EXF7(77010B)	•	SEL AND WAIT FOR DD65 MEM.	03K
		OUT(RHT2)	•		03K
03L	EXF(77211B)	SLJ(2EXT)	•	LIGHT OFF AND EXIT.	03K
	ENA(41B)	SLJ4(1PB)	•	PLACE INDICATORS ON RIGHT TUBE.	03K
	ENI1(35B)		•		03K
	LDA1(CHAI)	SST(MK11)	•		03K
	STAI(CHAI)	IJP1(L-1)	•		03K
	ENI1(27B)		•		03K
	LDA1(RM2)	SCL(MK11)	•		03K
	STAI(RM2)	IJP1(L-1)	•	SEL AND WAIT FOR DD65 MEM.	03K
	EXF(77010B)	EXF7(77010B)	•	OUTPUT.	03K
		OUT(LFT1)	•	SEL AND WAIT FOR DD65 MEM.	03K
	EXF(77010B)	EXF7(77010B)	•		03K
		OUT(LFT2)	•	LIGHT OFF AND EXIT.	03K
	EXF(77211B)	SLJ(2EXT)	•		04K
* INTENSITY - N=NORMAL, B=BRIGHT.			•	ROUTE NEXT HIT.	04K
04A	ENA(04C)	SLJ(01AE)	•	NORMAL.	04K
04C	ENI6(1)	INA(-45B)	•	BRIGHT.	04K
	AJP(04N)	INA(-15B)	•	NOTA.	04K
	AJP(04G)	SLJ(1EXT)	•		04K
04N	ENA(50B)	SLJ4(1PB)	•	SEL AND WAIT FOR DD65 MEM.	04K
	EXF(77010B)	EXF7(77010B)	•		04K
		OUT(NOR1)	•		04K
	EXF(77010B)	EXF7(77010B)	•	SEL AND WAIT FOR DD65 MEM.	04K
		OUT(NOR2)	•		04K
	EXF(77221B)	SLJ(2EXT)	•	INTENSITY OFF.	04K
	ENA(51B)	SLJ4(1PB)	•		04K
04G	EXF(77010B)	EXF7(77010B)	•	SEL AND WAIT FOR DD65 MEM.	04K
		OUT(BRT1)	•		04K

EXF(77010B)	EXF(77010B)	SEL AND WAIT FOR DD65 MEM.	04K
EXF(77221B)	OUT(BRT2)	INTENSITY OFF.	04K
* INCREMENT - R=RIGHT	SLJ(2EXT)	ROUTE NEXT HIT.	05K
C5A ENA(05C)	SLJ(01AE)	RIGHT.	05K
C5C ENI6(1)	INA(-51B)	DOWN.	05K
AJP(05R)	INA(-13B)	NOTA.	05K
AJP(05D)	SLJ(1EXT)		05K
ENA(60B)	SLJ4(1PB)		05K
EXF(77010B)	EXF(77010B)	SEL AND WAIT FOR DD65 MEM.	05K
EXF(77010B)	OUT(RIT1)	SEL AND WAIT FOR DD65 MEM.	05K
OUT(RIT2)	EXF(77010B)	INCR = -1 = RIGHT.	05K
STA(INCR)	ENA(-1)	EXIT.	05K
SLJ(2EXT)	EXF(77241B)		05K
ENA(61B)			05K
EXF(77010B)	SLJ4(1PB)	SEL AND WAIT FOR DD65 MEM.	05K
EXF(77010B)	EXF(77010B)	SEL AND WAIT FOR DD65 MEM.	05K
EXF(77010B)	OUT(RIT1)		05K
EXF(77010B)	EXF(77010B)	INCR = 1 = DOWN.	05K
ENA(1)	SLJ(05E)	+ OR - AND 2 OCT DIGITS.	11K
* LEFT MARGIN = P-USE	PIP	PIP.	11K
11A ENA(11C)	SLJ(01AE)	NOT - SIGN, JUMP.	11K
11C INA(-47B)	AJP(11PI)	NOT + SIGN, ERROR EXIT.	11K
INA(7)	AJP(1L+2)	+ PIPPER.	11K
ENA(11N)	SLJ(01AE)	USE ONLY POSITS / BY 4.	11K
ENA(-20B)	AJP(1EXT)	NEG. INPUT.	11K
ENA(11P)	SLJ(01AE)	POS. INPUT.	11K
LDA(PIPX)	ARS(2)	EXIT.	11K
ALS(2)	SLJ(11E)		11K
SCM(SEV)	SLJ4(1RET)		11K
STA(LMAR)	SLJ4(1RET)		11K
EXF(77303B)	SLJ4(1MAR)		11K
* RIGHT MARGIN (SEE LEFT	SLJ(2EXT)		12K
12A ENA(12C)	SLJ(01AE)		12K
12C INA(-47B)	AJP(12PI)		12K
INA(7)	AJP(1L+2)		12K
ENA(12N)	SLJ(01AE)		12K
ENA(-20B)	AJP(1EXT)		12K
ENA(12P)	SLJ(01AE)		12K
LDA(PIPX)	ARS(2)		12K
ALS(2)	SLJ(12E)	PIP. ONLY POSITS / BY 4.	12K
SCM(SEV)	SLJ4(1RET)		12K
STA(RMAR)	SLJ(12E)		12K
EXF(77305B)	SLJ4(1RET)		12K
	SLJ4(1MAR)		12K
	SLJ(2EXT)	EXIT.	12K

* TAB	SET - PIPPER IS USED (PIPX).				
13A	LDA(PIPX)	ARS(2)		NEXT TAB IS PIPPER X LOCATION.	13K
	ALS(2)	STA(T)		ONLY POSITS / BY 4 ARE USED.	13K
	ENI(0)			ZERO INDEX.	13K
	LDA1(TL1)	SUB(T)		OLD TAB, EXIT.	13K
	AJP(1PTL)	AJP3(L+3)		NEW TAB, TOO FAR TO RIGHT, JUMP.	13K
	LDA(T)	LDQI(TL1)		INSERT NEW TAB.	13K
	STA1(TL1)	STQ(T)			13K
	ISK1(L7B)	SLJ(L-4)			13K
	LDA(PIPX)	STA(TL16)			13K
	SLJ(1PTL)			IN ANY EVENT STORE IN TL16.	13K
* TAB	CLEAR - PIPPER IS USED.				13K
14A	ENI(0)				14K
	LDA(PIPX)	SUB1(TL1)			14K
	AJP3(L+2)	LDA(PIPX)		TRY NEXT TAB IF NEG.	14K
	SUB1(TL2)	AJP3(14CL)		DELETE IF ONLY ONE.	14K
	ISK1(17B)	SLJ(L-3)			14K
	LDA(1PTL)			NONE FOUND.	14K
14CL	LDA1(TL2)	STA1(TL1)		EXTRACT EXTRA TAB.	14K
	ISK1(17B)	SLJ(L-1)			14K
	SLJ(1PTL)				14K
* SPACING	ENI(0)				15K
15A	LDA(15C)	SLJ(01AE)			15K
15C	INA(-22B)	AJP(15S)		SINGLE.	15K
	INA(-42B)	AJP(15D)		DOUBLE.	15K
	INA(41B)	AJP(15T)		TRIPLE.	15K
	SLJ(1EXT)			NO TA.	15K
15S	ENA(1)	STA(SPAC)		DELTA = SPACING * SIZE (528).	15K
	EXF(77010B)	EXF7(77010B)		SEL AND WAIT FOR DD65 MEM.	15K
	OUT(SG1)	SLJ(15E)			15K
15D	ENA(2)	STA(SPAC)			15K
	EXF(77010B)	EXF7(77010B)		SEL AND WAIT FOR DD65 MEM.	15K
	OUT(DB1)	SLJ(15E)			15K
15T	ENA(3)	STA(SPAC)			15K
	EXF(77010B)	EXF7(77010B)		SEL AND WAIT FOR DD65 MEM.	15K
	OUT(TP1)	SLJ(15E)			15K
15E	ENA(4)	EXF7(77010B)		SEL AND WAIT FOR DD65 MEM.	15K
	OUT(SG2)	SLJ4(15Z)		RECAL SPACING.	15K
	EXF(77341B)	SLJ(2EXT)		EXIT.	15K
* TYPE	ENA(0)				31K
31A	LDA(MCDE)				31K
	INA(-2)	AJP(L+3)		CHAR.	31K
	AJP(31SA)	AJP3(31V)		VECTOR.	31K
	ENA(31CA)	SLJ(1EXT)		SPOIT. CHAR.	31K
31C1	SAU(1STP)			POSITION.	31K
	ENA(0)	SLJ4(1LOC)		START STRING.	31K
		SLJ4(1PB)		MARKER	31K
		SLJ4(33MK)			31K
31V	SLJ(OEXT)				31K
	ENA(31VA)	SAU(1STP)		ROUTE VECTOR.	31K


```

* MAIN CHARACTER LOOP.
31CA  SLJ(31CI)
      AJP(1BKS)
      AJP(31CT)
      AJP(31CR)
      AJP(OEXT)
      ENI6(0)

31CT  SLJ(OEXT)
      ENI1(0)
      LDA(INCR)
      LDA1(TL1)
      SUB(XLOC)
      ISK1(17B)
      SLJ(1EXT)
      SUB(SIZE)
      ENA(20B)

31CT1 SLJ(31CT)
      LDA(YLOC)
      SUB1(TL1)
      ISK1(17B)
      IJP(31CT+7)
      SLJ(N)
      LDQ(INCR)
      RAD(XLOC)
      AJP3(31CM)
      RSB(YLOC)
      STA(XLOC)
      ENA(77B)

31CM  ENI6(0)
      ENA(0)
      SLJ(OEXT)
      RSB(YLOC)
      AJP2(31CM)
      STA(YLOC)
      ADD(XLOC)
      LAC(INCR)
      AJP3(31CM+1)
      * VECTOR - S=SMALL, 3=SE, 4=SW, 5=LARGE,
      * 9=NE, 6=E, 7=MEDIUM, 8=NORTH,
      * 31VA  AJP(1EXT)
      AJP(31VU)
      AJP(31VB)
      AJP(31VS)
      AJP(31VL)

      INA(-36B)
      INA(-40B)
      INA(-1)
      INA(77B)
      SLJ4(1PB)
      SLJ4(31CM)
      SLJ4(33MK)

      ENI6(0)
      AJP2(31CT1)
      ADD(SIZE2)
      AJP2(L+3)
      SLJ(L-2)

      AJP3(31CA+6)
      SLJ4(1PB)
      SLJ4(31CM)

      SUB(SIZE2)
      AJP3(L+3)
      SLJ(L-2)

      SLJ(31CR)
      LDA(SIZE)
      QJP2(31CM1)
      SUB(RMAR)
      LDA(DELTA)
      LDA(LMAR)
      ENI6(2)
      SLJ4(1PB)
      SLJ4(1LOC)
      SLJ4(33MK)
      SLJ4(1PB)

      INA(377B)
      ENA(376B)
      LDA(DELTA)
      SLJ(31CM+5)

      SLJ(31CM+3)
      * E=EXTRA LARGE, 0=BLANK, 5=UNBLANK
      * 7=NW, 8=NORTH,
      * 0, EXIT,
      * UNBLANK, JUMP.
      * BLANK.
      * SMALL.
      * LARGE.

```


[illegible]

STA(YLOC)	STQ(XLOC)	.		32K
SLJ(32T2)	CHAR INCLUDING SPACE.	.		32K
* MARKER - ANY KB2	SLJ(OIAE)	.	ROUTE.	33K
33A ENA(33C)		.	POSITION CHARACTER.	33K
33C ALS(368)	SSU(MARKER)	.		33K
LDQ(MKA)	SLJ4(33MX)	.	ADDRESS MARKER.	33K
STA(MARKER)	SLJ(2EXT)	.		33K
EXF(775118)		.		33K
* CLEAR PICBUF	STA(POSIT)	.	ZEROIZE POSIT.	34K
34A LDA(INPOSIT)	ENI1(1000B)	.	CLEAR PICBUF.	34K
ENA(O)	IJPI(L)	.		34K
STAT(PICBUF)		.		34K
ENI1(1000B)	EXF7(77010B)	.	CLEAR DD65 MEM.	34K
EXF(77010B)	IJPI(L-1)	.	SEL AND WAIT FOR DD65 MEM.	34K
OUT(ERC2)	ENI6(1)	.		34K
	SLJ4(1PB)	.	START FILE.	34K
ENA(O1)		.	LIGHTS OFF.	34K
SLJ(55A)		.		35K
* TEST KEYBOARD 2 LIGHTS.	SLJ4(FLASH)	.	EXIT VIA RESET.	35K
35A EXF(77541B)		.		35K
SLJ(55A)		.		41K
* PIP ON/OFF.	ENQ(1)	.	POS = ON, NEG = OFF.	41K
41A LDA(PIPQ)	ENQ(-1)	.	PIPQ NEG., TURN OFF.	41K
AJP3(L+1)	SLJ(1PIP)	.	POS, TURN ON.	42K
STQ(PIPQ)		.		42K
* TRACK BALL.	INT(PIPX)	.		42K
42A EXF(77102B)	INT(PIPY)	.		44K
EXF(77104B)	SLJ(1PIP)	.		44K
	CHARACTER EDIT - R=REPLACE WITH NEXT CHAR, D=DELETE AND CLOSE UP, FLASHING=CHAR NOT FOUND.	.		44K
* I=INSERT NEXT CHAR TO THE LEFT.	SAU(1STP)	.		44K
* 44A ENA(44C)	SAL(C1+1)	.		44K
ENA(PICBUF)	SAL(C1+2)	.		44K
ENA(POSIT)	SAL(C1+3)	.		44K
ENA(HERE)	SAL(C1+4)	.		44K
ENA(PIPX)	SAL(C1+5)	.		44K
ENA(PIPY)	SLJ4(C1)	.	DO CHALOC.	44K
	ENI2(6)	.		44K
AJP2(OEXT)	SLJ4(3WAT)	.	ON.	44K
EXF(77620B)	SLJ4(3WAT)	.	OFF.	44K
EXF(77621B)	SLJ4(OEXT)	.	BLINK.	44K
IJP2(L-2)	AJP(44R)	.	REPLACE.	44K
INA(-13B)	AJP(44D)	.	DELETE.	44K
INA(-5B)	AJP(44I)	.	INSERT.	44K
INA(-5B)		.	NOTA, ERROR EXIT.	44K
SLJ(1EXT)		.		44K
* REPLACE CHAR WITH ANY KB1 CHAR.		.		44K
44R ENA(44R1)	SLJ(OIAE)	.	ROUTE NEXT BYTE BELOW.	44K
44R1 LIL(HERE)	STA(TEMP)	.	CHAR TO TEMP.	44K


```

ENA(PIPX)
ENA(PIPY)
AJP2(OEXT)
EXF(77720B)
EXF(77721B)
IJP2(L-2)
INA(-13B)
SLJ(1EXT)
* REPLACE
54R ENA(L+1)
LIL1(HERE)
AJP(OEXT)
AJP(OEXT)
AJP2(54RX)
STAI( )
LRS2(6)
LLS(6)
LDA1(PICBUF)
LRS2(6)
ENA(PICBUF)
IJP2(L+2)
RAO(HERE)
INI2(-5)
SLJ(OEXT)
EXF(77721B)
* DELETE AND CLOSE UP
54R ENA(L-1)
LIL1(HERE)
LDA1(PICBUF)
ENI3(7)
ENQ(0)
QJP(54DF)
IJP1(L+1)
LDA1(PICBUF)
LDA1(10B)
LDA1(-1)
LDA1(PICBUF)
LDA4(KO)
AJP(54DA)
SST4(KO)
IJP4(L-3)
ISK1(1000B)
* MOVE EACH LINE UP
54DA LIL1(HERE)
LDA1(PICBUF)
SAL(C1+4)
SAL(C1+5)
SLJ4(C1)
SLJ2(3WAI)
SLJ4(3WAI)
SLJ2(2EXT)
SLJ(54R)
AJP(54D)
* DO CHALOC° BLINK KEY.
* NOT HERE° BLINK KEY.
* ON°
* OFF°
* DELETE AND REPLACE°
* DELETE AND CLOSE UP°
* NOT°
* ROUTE NEXT HIT BELOW°
* RELOAD INDICIES°
* IGNORE BACKSPACE AND TAB°
* EXIT ON CR°
* OLD BYTE IN Q(LJ)°
* END OF STRING° EXIT°
* NEW BYTE IN Q(LJ)°
* NEW WORD FORMED AND STORED°
* PRINT
* NOT LAST BYTE IN WORD° JMP°
* CALL NEXT WORD°
* STORE INDICIES°
* LIGHT OFF AND EXIT°
* FIND THE START OF THE STRING(00)°
* X BYTE IN Q(LJ)°
* X BYTE AND ALL TO THE RIGHT = 77°
* *** EXAM BYTES FROM RT TO LF°
* *** 00 FOUND° JUMP°
* *** PICBUF EXHAUSTED° JUMP°
*
* B4 = 7 - B3°
* *** REPLACE ALL BYTES IN THE
* *** STRING WITH 77°
* *** END OF STRING° JUMP°
* ***
* ***
* ***

```



```

STQ(W)
ENQ(O)
DVI(M6)
ENI2(N)
ARS(2)
SCM(SEV)
INA(2100B)
STA(TNY)
SAU(54DC)
SAU(54DB)
54DI READ NEXT BYTE.
54DR ISK1(1000B)
SLJ(L+4)
LDA(PICBUF)
LDQ(W)
LLS(6)
BRANCH POINT.
54DB SLJ(N)
54DS SKIP STRING.
54DS INA(-77B)
54DI SLJ(54D1)
54DC DECODER.
54DP AJP(54DO)
54DP AJP3(54DR)
54DP AJP2(L+2)
54DP QJP3(54DX)
54DP INA(-53B)
54DP SLJ(54DR+2)
54DP PROCESS BEGIN STRING.
54DO LDA(TNY)
54DP ENA(54DS)
54DP PROCESS X LOCATION.
54DX ENA(L+1)
54DI SLJ(54D1)
54DP PROCESS Y LOCATION.
54DY LLS(1)
54DP ALS(6)
54DP ENA(L+1)
54DP RAD(TNY)
54DP AJP(L+6)
54DP MUI(M6)
54DP SAU(L+2)
54DP ENQ(77000B)
54DP QLS(N)
54DP STAT(PICBUF)
54DP LDQ1(PICBUF)
54DP LDA(PNY)
54DP STQ1(PICBUF)
ENA2{O}
SAU(L+1)
LDA(PIPY)
AJP2(L+2)
INA(200B)
STA(PNY)
ENI2(7)
SLJ(L+2)
STA(W)
ENA(O)
STQ(W)
AJPI(54DR)
INA(-20B)
INA(-4)
LRJ(1)
SLJ(54DY)
AJPI(54DR)
SLJ(54DR+2)
STA(PNY)
SLJ(54D1+1)
SLJ(54D1+1)
INA(248)
STA(TNY)
SLJ(54D1+1)
ENA2(-7)
INA(7)
SAU(L+2)
LDA(PNY)
ALS(N)
SSU1(PICBUF)
SLJ(54D1)
LLS(6)
LRS(6)
LRS(6)
CAL PERMINENT NEW Y.
PIPY TO PICGEN CODE.
PICBUF EXAUSTED, EXIT.
NOT END OF STRING, EXIT.
BEGIN STRING, JUMP.
LESS THAN 20, READ NEXT BYTE.
NOT A LOC, JUMP.
20 + 22 TO X, 21 + 23 TO Y.
NOT END OF FILE, EXIT.
FINAL EXIT.
TEMP BECOMES PERM YLOC.
SKIP STRING.
SKIP NEXT BYTE.
RESTORE NORMAL ROUTE.
EXTRACT THIS Y FOR USE ON NEXT LINE.
```


STA(TIM)	SLJ(2EXT)	°	64K
RAO(TIM)	RSB(PIPX)	°	64K
SLJ(IPIP)		°	64K

* INTER-KEY SUBROUTINES.

* PRINT TAB LIST.			
1PTL ENI(0)	LDQ(MKX)	°	PTK
LDA(1TL1)	ALS(44B)	°	PTK
SSU(1TB1)	STA(T)	°	PTK
EXF(77010B)	EXF7(77010B)	°	PTK
	OUT(T)	°	PTK
ISK1(17B)	SLJ(L-4)	°	PTK
EXF(77311B)	EXF(77321B)	°	PTK
DELTA.		°	PTK
* CAL	LDA(SPAC)	°	DLK
15Z MUI(SIZE)	STA(DELTA)	°	DLK
* PIP		°	DLK
1PIP INCREMENTATION.	AJP3(2PIP)	°	PIK
LDA(PIPIQ)	LDQ(PIPY)	°	PIK
LDA(PIPX)	ARS(47B)	°	PIK
QRS(47B)	STA(PIPX)	°	PIK
STQ(PIPY)	QLS(14B)	°	PIK
LLS(44B)	LDQ(MKXY)	°	PIK
SSU(PIPI)	STA(PIPI)	°	PIK
EXF(77010B)	EXF7(77010B)	°	PIK
	OUT(PIPI)	°	PIK
EXF7(77174B)	SLJ(L+3)	°	PIK
ENI(1)	ISK1(10000B)	°	PIK
EXF7(77175B)	SLJ(2EXT)	°	PIK
EXF(77120B)	INT(T)	°	PIK
SLJ(L-4)		°	PIK
2PIP EXF(77010B)	EXF7(77010B)	°	PIK
	OUT(PIPOFF)	°	PIK
EXF(77603B)	SLJ(2EXT)	°	PIK
* CONVERT A SIGNED	BIT RJ NR TO A	°	PIK
1VRT SLJ(N)	ENQ(6000B)	°	PIK
	SCM(SEV)	°	PIK
AJP2(L+2)		°	PIK
ENQ(4000B)	LR(10B)	°	PIK
STQ(TEMP)	LLS(3)	°	PIK
ENA(0)	ENA(0)	°	PIK
RAD(TEMP)	AJP1(L+2)	°	PIK
LLS(3)		°	PIK
ENA(12B)	LDA(TEMP)	°	PIK
LR(6)		°	PIK

LOAD X POSIT MASK			
POSITION X			
SEL AND WAIT FOR DD65 MEM.			
OUTPUT.			
LIGHTS OUT.			
DELTA = SPACING * SIZE.			
IF PIP IS OFF, JUMP.			
EXTEND SIGN.			
POSITION ADDRESSES.			
INSERT NEW ADDRESSES.			
SEL AND WAIT FOR DD65 MEM.			
PRINT.			
KB2 HIT, JUMP.			
WAIT.			
STILL NOT HIT, RETURN.			
DUMP INPUT.			
LOOP.			
SEL AND WAIT FOR DD65 MEM.			
TURN OFF PIP (KB2).			
SIGN + 2 BYTE BCD OCT NR.			
BCD +.			
IF NEG, COMPLEMENT.			
BCD --.			


```

* PLACE XLOC AND YLOC SLJ(1VRT)
1LOC SLJ(N) ENI6(10B)
LDA(XLOC) LDQ(YLOC)
LDS(47B) LRS(47B)
QRS(47B) STA(XLOC)
STQ(YLOC) AJP2(L+3)
ENAC(22B) SLJ4(1PB)
LAC(XLOC) SLJ(L+3)
ENA(20B) SLJ4(1PB)
LDA(XLOC)
ARS(2) SLJ4(1PB)
LDA(YLOC) SLJ4(1PB)
ENA(23B) SLJ4(1PB)
LAC(YLOC) SLJ(L+3)
ENA(21B) SLJ4(1PB)
LDA(YLOC)
ARS(2) SLJ4(1PB)
SLJ(1LOC)
* LOADER FOR PICBUF.
1PB SLJ(N)
SIU2(6PB)
LIU2(POSIT)
LRS(6)
ARS2(0)
STA(0) PICBUF
ENA2(-6)
ISK1(777B)
ENI6(3)
OUT6(FUL3)
ENI2(66B)
INI2(-6)
ENA(PICBUF)
2PB
4PB
3PB
6PB
SLI1(POSIT)
ENI2(N)
ENA(0)
* WAIT
1WAT SLJ(N)
IJP1(L)
3WAT SLJ(N)
AJP(-1)
INA(-1) AND RIGHT MARGINS.
* PRINT SLJ(N)
1MAR INAC(3)
LDQ(RMX)
STA(RM1)
INA(-3)

```


SSU(LM1)	STA(LM1)	•		MRK
LDA(RMAR)	SLJ4(1VRT)	•		MRK
ALS(14B)	LDQ(MKN)	•		MRK
SSU(RM2)	STA(RM2)	•	X TO RM2.	MRK
LDA(LMAR)	SLJ4(1VRT)	•		MRK
ALS(14B)	LDQ(MKN)	•		MRK
SSU(LM2)	STA(LM2)	•	X TO LM2.	MRK
ENI(4)		•		MRK
EXF(77010B)	EXF7(77010B)	•	SEL AND WAIT FOR DD65 MEM.	MRK
OUT1(RM2)	LDA(RMAR)	•		MRK
SUB(LMAR)	ENQ(0)	•		MRK
AJP2(L+1)	ENA(0)	•	NO NEG. THAN 100	MRK
DVI(SIZE)	INA(-144B)	•	GREATER YES, JUMP.	MRK
AJP2(L+2)	INA(144B)	•		MRK
ENQ(0)	SLJ(L+2)	•		MRK
ENQ(100B)		•		MRK
STQ(T)	ENI(0)	•	HOW MANY TENS.	MRK
INA(-12B)	AJP3(L+2)	•		MRK
INI(1)	SLJ(L-1)	•		MRK
INA(12B)	AJP1(L+2)	•		MRK
INA(12B)		•		MRK
LRS(6)	ENA(0)	•	ZERO = 12 IN BCD.	MRK
AJP1(L+1)	ENA(12B)	•		MRK
ADD(T)	LLS(36B)	•		MRK
LDQ(MKA)		•		MRK
SSU(CPL3)	STA(CPL3)	•		MRK
ENI(3)		•	OUTPUT TO DESPLAY.	MRK
EXF(77010B)	EXF7(77010B)	•	SEL AND WAIT FOR DD65 MEM.	MRK
OUT1(CPL3)	SLJ(1MAR)	•		MRK
* NEXT 2 HITS CONVERTED	ED TO BINARY.	•		MRK
1RET	INA(-12B)	•		RTK
SLJ(N)	INA(2)	•	BCD 0, JUMP.	RTK
AJP(L+2)	INA(10B)	•	NON-OCTAL, EXIT.	RTK
AJP2(1EXT)	STA(T)	•	POSITION 1ST BYTE.	RTK
ALS(5)	SLJ(01AE)	•	ROUTE LAST HIT.	RTK
ENA(L+1)	AJP(L+3)	•		RTK
INA(-12B)	AJP(1EXT)	•		RTK
INA(2)	ALS(2)	•		RTK
INA(10B)	SCL(MKY)	•		RTK
ADD(T)	SLJ(1RET)	•		RTK
* PRINT MARKER.		•		MKK
33MK	LDA(XLOC)	•		MKK
SLJ(N)	QLS(14B)	•	POSITION X + Y. LOAD MASK.	MKK
LDQ(YLOC)	LDQ(MKXY)	•		MKK
LLS(44B)	STA(MARKER)	•		MKK
SSU(MARKER)	EXF7(77010B)	•	SEL AND WAIT FOR DD65 MEM.	MKK
EXF(77010B)	SLJ(33MK)	•		MKK
OUT(MARKER)		•		BSK
* BACKSPACE	LDA(SIZE)	•	BACKSPACE MARKER.	BSK
1BKS		•		BSK

1RNB	IJP1(L+5)	ENI1(7B)	• READ NEXT BYTE.	RBP
	ISK2(1000B)	SLJ(2RNB)	• ANY WORDS LEFT.	RBP
	SLJ(1FN)		• NO EXIT.	RBP
2RNB	ZRO(0)	ZRO(0)	• WORD	RBP
	LDA2(N)	STA(L-1)	• LOAD NEW WORD.	RBP
	LDQ(L-2)	ENA(0)	• WORD TO Q, 0 TO A.	RBP
	LLS(6B)	STQ(L-3)	• LOAD NEXT BYTE, Q TO WORD.	RBP
* MAIN BRANCH POINT.			• MAIN BRANCH POINT.	BPP
1BRA	SLJ(N)			BPP
* PROCESS STRING.				PSP
1CST	INA(-76B)	AJP2(1EST)	• IF BYTE IS CR OR END STRING, JUMP.	PSP
	INA(76B)	SLJ4(1PAK)	• PACK.	PSP
	SLJ(1RNB)			PSP
1SPT	ALS(6)	STA(W1)	• 1ST BYTE.	PSP
	ENA(L+1)	SLJ(4ST)	• 2ND BYTE.	PSP
	ADD(W1)	ALS(6)		PSP
	STA(W1)			PSP
	ENA(L+1)	SLJ(4ST)	• 3RD AND LAST BYTE.	PSP
	ADD(W1)	LLS(11B)	• CDOOR ARE IN POSITION.	PSP
	ALS(33B)	LLS(11B)	• INSERT LOCATION.	PSP
	LDQ(MKXY)	SSU(DW)		PSP
	STA(W1)	LDQ(MKPDW)		PSP
	LDA(MKSPT)	SSU(W1)	• INSERT DOT.	PSP
	STA(W1)	INI3(1)	• INDEX MEM.	PSP
	EXF(77010B)	EXF7(77010B)	• SEL AND WAIT FOR DD65 MEM.	PSP
	OUT(W1)	SLJ(3ST)	• OUTPUT.	PSP
* END	STRING.			ESP
1EST	ENA4(-6)	AJP(3ST)	• OUTPUT WORD COMPLETE, JUMP.	ESP
	ENA(20B)	SLJ4(1PAK)	• FILL WITH BLANKS.	ESP
2EST	SLJ(1EST)			ESP
* PACK				PKP
1PAK	SLJ(N)	RAD(W)		PKP
2PAK	IJP4(L+1)	SLJ(N)	• WORD FULL, JUMP TO 4 OR 5 PAK	PKP
	ALS(6)	STA(W)	• NOT FULL, STORE AND RETURN.	PKP
4PAK	SLJ(1PAK)			PKP
	ENA(5PAK)	SAL(2PAK)	• DW.	PKP
	LDA(W)			PKP
	ALS(30B)	LDQ(MKPDW)		PKP
5PAK	SSU(DW)	SLJ(L+4)		PKP
	LLS(6B)	ALS(6B)		PKP
	LLS(6B)	ENQ(07700B)	• SW STRING.	PKP
	SSU(SW)			PKP
	ISK3(727B)	SLJ(L+2)	• UPDATE MEM ADDRESS.	PKP
	SLJ(1ER)	EXF7(77010B)	• AVAILABLE MEM FILLED, EXIT.	PKP
	EXF(77010B)	STA(W)	• SEL AND WAIT FOR DD65 MEM.	PKP
	ENI4(6)	ENA(0)	• NEW WORD.	PKP
	OUT(W)	SLJ(1PAK)		PKP
* TABLE	STA(W)			TBP

10G	INA(00D)	SAU(L+1)	ENTER TABLE.	TBP
00D	SLJ(N)	SLJ(1BGS)	JUMP.	TBP
	ENA3(0)	SLJ(1ERR)	BEGIN STRING.	TBP
	SLJ(1RNB)	SLJ(1ERR)	PASS.	TBP
	ENA(2ERR)	SLJ(1ERR)		TBP
	ENA(2ERR)	SLJ(1ERR)		TBP
	ENA(2ERR)	SLJ(1ERR)		TBP
	ENA(2ERR)	SLJ(1ERR)		TBP
10D	ENA(1CST)	SLJ(1CH)	CHARACTER STRING.	TBP
	ENA(1CST)	SLJ(1VCT)	VECTOR STRING.	TBP
	ENA(1SPT)	SLJ(1CH)	SPOT MODE.	TBP
	ENA(2ERR)	SLJ(1ERR)		TBP
	ENA(2ERR)	SLJ(1ERR)		TBP
	ENA(2ERR)	SLJ(1ERR)		TBP
	ENA(2ERR)	SLJ(1ERR)		TBP
20D	ENA(20AD)	SLJ(1ERR)		TBP
	ENA(21AD)	SLJ(4ST)	SET X.	TBP
	ENA(22AD)	SLJ(4ST)	SET Y.	TBP
	ENA(23AD)	SLJ(4ST)	SET -X.	TBP
	ENA(2ERR)	SLJ(1ERR)	SET -Y.	TBP
	ENA(2ERR)	SLJ(1ERR)		TBP
	ENA(2ERR)	SLJ(1ERR)		TBP
30D	ENA(2ERR)	SLJ(1ERR)		TBP
	LDA(DW)	SLJ(1SML)	SET SMALL CHAR.	TBP
	LDA(DW)	SLJ(1MED)	SET MEDIUM CHAR.	TBP
	ENA(2ERR)	SLJ(1LRG)	SET LARGE CHAR.	TBP
	ENA(2ERR)	SLJ(1ERR)		TBP
	ENA(2ERR)	SLJ(1ERR)		TBP
	ENA(2ERR)	SLJ(1ERR)		TBP
	ENA(2ERR)	SLJ(1ERR)		TBP
40D	ENA(2ERR)	SLJ(1ERR)		TBP
	LDA(DW)	SLJ(1ERR)	USE RIGHT TUBE.	TBP
	LDA(DW)	SLJ(1TRT)	USE LEFT TUBE.	TBP
	ENA(2ERR)	SLJ(1ERR)		TBP
	ENA(2ERR)	SLJ(1ERR)		TBP
	ENA(2ERR)	SLJ(1ERR)		TBP
	ENA(2ERR)	SLJ(1ERR)		TBP
50D	ENA(2ERR)	SLJ(1ERR)		TBP
	LDA(DW)	SLJ(1ERR)	SET NORMAL INTENSITY.	TBP
	LDA(DW)	SLJ(1NOR)	SET BRIGHT INTENSITY.	TBP
	ENA(2ERR)	SLJ(1BRT)		TBP
	ENA(2ERR)	SLJ(1ERR)		TBP
	ENA(2ERR)	SLJ(1ERR)		TBP
	ENA(2ERR)	SLJ(1ERR)		TBP

60D	EN A(2ERR)	SLJ(1ERR)	TBP
	LDA(DW)	SLJ(1INR)	TBP
	LDA(DW)	SLJ(1IND)	TBP
	EN A(2ERR)	SLJ(1ERR)	TBP
	EN A(2ERR)	SLJ(1ERR)	TBP
	EN A(2ERR)	SLJ(1ERR)	TBP
	EN A(2ERR)	SLJ(1ERR)	TBP
	EN A(2ERR)	SLJ(1ERR)	TBP
	EN A(2ERR)	SLJ(1ERR)	TBP
	EN A(2ERR)	SLJ(1ERR)	TBP
	EN A(2ERR)	SLJ(1ERR)	TBP
	EN A(2ERR)	SLJ(1ERR)	TBP
	EN A(2ERR)	SLJ(1ERR)	TBP
76D	SLJ(1FN)		TBP
77D	SLJ(1RN B)		TBP
* 1BGS	* BEGIN STRING.		TBP
	AL S(15B)		TBP
28GS	SU U(4PAK)		BSP
	EN I4(1)		BSP
	SL J(4ST)		BSP
* 1CH	* MODE		MDP
	AL (2BGS)	LDA(DW)	MDP
	SCL(MKCH)	STA(DW)	MDP
	LDA(SW)	SCL(MKCH)	MDP
	STA(SW)	SLJ(1RN B)	MDP
1VCT	AL (2BGS)	LDA(DW)	MDP
	SST(MKCH)	STA(DW)	MDP
	LDA(SW)	SST(MKCH)	MDP
	STA(SW)	SLJ(1RN B)	MDP
* 1LOCAT ION.			LCP
22AD	SCM(MKCM)		LCP
20AD	ALS(46B)	LDQ(MKX)	LCP
	SU U(DW)	SLJ(L+4)	LCP
23AD	SCM(MKCM)		LCP
21AD	ALS(2B)	LDQ(MKY)	LCP
	SU U(DW)		LCP
	STA(DW)	SLJ(3ST)	LCP
* 1SML	SIZE		SZP
	SCL(MK47)		SZP
1MED	STA(DW)	SCL(MK46)	SZP
	SCL(MK47)	SLJ(1RN B)	SZP
	STA(DW)	SST(MK46)	SZP
1LRG	STA(DW)	SLJ(1RN B)	SZP
	SST(MK47)	SCL(MK46)	SZP
	STA(DW)	SLJ(1RN B)	SZP
* 1TRT	SST(MK11)	STA(DW)	TBP
			TBP

[illegible]

```

MACHINE CHALOC (PICBUF, POSIT, HERE, PIPX, PIPY)
FINDS THE LOCATION IN PICBUF OF THE CHARACTER INDICATED BY THE PIPER.
HERE (LA) = PICBUF WORD AND HERE(UA) = BYTE (RT MOST=0, LF MOST=52B)
CON(M6 =6B)
SLJ(N)
ZRO(0)
ZRO(0)
ZRO(0)
ZRO(0)
ZRO(0)
LDA7(3ST)
ENAI(1SF)
LDAI(2ST)
ENI(0)
READ NEXT BYTE.
IRNB IJPI(L+4)
*
1ST
2ST
3ST
4ST
5ST
6ST
*
SLJ(L+6)
ZRO(0)
ZRO(0)
ZRO(0)
ZRO(0)
ZRO(0)
SAU(5RNB)
SAU(1BRA)
SAU(3RNB)
ENI2(0)
ENI(7)
*
PICBUF.
POSIT. IF HERE IS NEG. NOT FOUND.
PIPX.
PIPY.
*
PREP. B2= WORD.
BI=BYTE
USE OLD WORD, JUMP.

```


[illegible]

thesG825

Programming a remote monitor and display



3 2768 002 13928 9

DUDLEY KNOX LIBRARY